Sonification and Visualization of Parametric Equations

Edward Ball
Academo.org
London, UK
info@academo.org

ABSTRACT

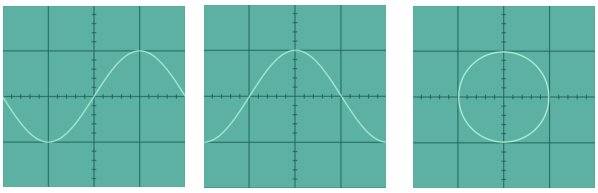
This paper describes how parametric equations can be sonified using the Web Audio API and visualized using a vectorscope built using the HTML5 canvas element. A working demonstration can be found at academo.org/demos/vectorscope, with a selection of user-adjustable parametric equations, including Lissajous figures and hypotrochoids.

1. INTRODUCTION

Parametric equations are sets of equations that express quantities in terms of independent variables known as "parameters". An example of a pair of such equations would be

$$x = \sin t \tag{1a}$$
$$y = \cos t \tag{1b}$$

In this case our parameter is t. Plots of x against t, and y against t, over the range $t = -\pi$ to $t = +\pi$, are shown in Figures 1a and 1b. Figure 1c shows a plot of x against y over the same range - this method of visualizing parametric equations is what will be used throughout this paper. If instead of $y = \cos t$, equation 1b was $y = \sin t$, Figure 1c would simply be a diagonal line from bottom left to top right, since at all times x would be equal to y.



(a) (b) (c)

Figure 1: Plots of equations 1a, 1b, and x against y over the range over the range $t = -\pi$ to $t = +\pi$

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

2. SONIFICATION AND VISUALIZATION

2.1 Sonification

Sonifying parametric equations is a matter of creating two channels of audio, one for the x variable and one for the y. Equations 1a and 1b can be sonified using two Web Audio oscillator nodes, as long as the phase difference between the two oscillators is $\pi/2$. At the time of writing, phase control of oscillator nodes is unavailable in Web Audio but is in development. However, we can still create a phase offset by using a `delayNode` with its `delayTime` set to one quarter of a period of the current frequency.

For more complicated parametric equations, an alternative approach is to create a stereo `AudioBufferSourceNode` and fill the left and right buffers with data that has been calculated programatically using the parametric equation for that channel. This is done over a certain range of t (ideally one which contains a whole number of periods which therefore avoids discontinuities and audio artefacts) and the `loop` property of the buffer is set to true.

2.2 Visualization

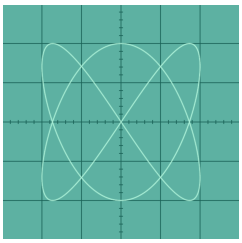
Visualizing parametric equations can be done using the HTML5 canvas drawing API. The displacement at a specific time can be obtained via the `getFloatTimeDomainData` method of a Web Audio `analyserNode`. However, at the time of writing, the native Web Audio analyser node does not support stereo input, so we utilise an MIT licensed custom implementation [1]. The two audio channels are passed to the analyser which then fills two arrays with the time domain data.

With appropriate scaling, the horizontal offset on the canvas is determined by the time domain data from one channel, the vertical offset by the data from the other channel and the data is plotted as in Figure 1c. This process is repeated at approximately 60fps using the `requestAnimationFrame` method allowing any time evolution of the pattern to be shown on the screen.

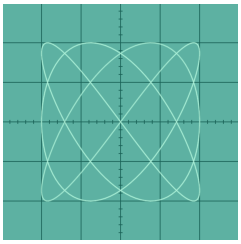
Pieces of hardware that plot such x-y graphs of signals are known as *vectorscopes*, a special type of oscilloscope. Users familiar with standard oscilloscopes can also recreate this functionality by enabling "x-y mode" on their particular model.

3. RESULTS

3.1 Lissajous Figures

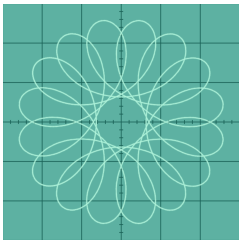


(a) 400 Hz and 600 Hz

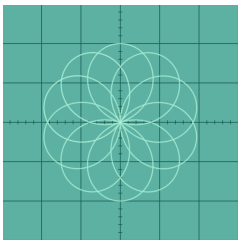


(b) 450 Hz and 600 Hz

Figure 2: A selection of Lissajous figures.



(a) Hypotrochoid



(b) Rose

Figure 3: An example hypotrochoid and rose.

In their most general form, Lissajous Figures are created from the following pair of parametric equations [3].

$$x = A\sin(at + \delta) \tag{2a}$$
$$y = B\sin(bt) \tag{2b}$$

For simplicity, our demo has $A = B = 1$. Two oscillator nodes are created, with adjustable frequencies determined by a and b. The relative phase offset between them would be δ. Due to the lack of precise phase control as mentioned earlier, the δ slider in the demo doesn't represent an accurate phase offset (as the oscillator's relative phase changes when the frequency changes) but nevertheless still provides a way to nudge the phase and adjust the visualization.

Lissajous patterns can range from very simple to very intricate depending on the ratio of $a : b$. In terms of sound, they are very simple - two pure tones. A selection of Lissajous patterns and their settings is shown in Figure 2.

3.2 Hypotrochoids

Mathematically speaking, hypotrochoids are created via the following pair of parametric equations [2].

$$x(t) = (R - r)\cos(t) + d\cos\left(\frac{R - r}{r}t\right) \tag{3a}$$
$$y(t) = (R - r)\sin(t) - d\sin\left(\frac{R - r}{r}t\right) \tag{3b}$$

Hypotrochoids are made up of 4 sinusoids, so again they can be created simply using web audio oscillators. Sonically, they sound very similar to Lissajous figures, as they consist of just two pure frequencies. Despite the simple sound, they can produce some very intricate patterns such as that in Figure 3a.

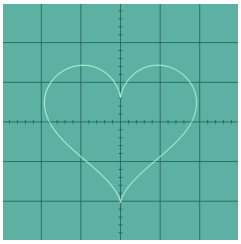
3.3 Rose Curves

Rose curves are created by equations 4a and 4b [4]. Since these equations involve multiplication of sinusoids, we are able to recreate these in Web Audio by connecting oscillators to `gainNodes` that have their gain value modulated by a separate oscillator.

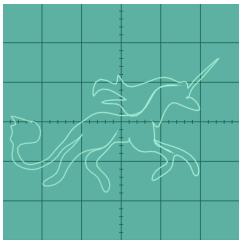
$$x(t) = \cos(kt)\cos(t) \tag{4a}$$
$$y(t) = \cos(kt)\sin(t) \tag{4b}$$

3.4 Pre-calculated curves

Producing more complex plots can be done by adding large numbers of sinusoids together. While it should be possible to recreate these by creating large numbers of corresponding oscillator nodes, the author has for the following cases decided to create an empty buffer and programatically caluclate the required samples over a single period. Two such examples are shown in Figure 4.



(a) Heart shape



(b) Unicorn shape

Figure 4: Plots generated by programatically filling a buffer.

To introduce time evolution, this process can be repeated over a number of periods. For each period, a transformation is applied to the samples. The transformation is then adjusted slightly for the next iteration. In this way, the author was able to create a spinning ballerina, and a jumping dolphin.

4. CONCLUSIONS

Using combinations of `oscillatorNodes`, `gainNodes`, `delayNodes` and `bufferSourceNodes` a wide range of interesting parametric equations can succesfully be sonified using the Web Audio API and visualized on a custom canvas-based vectorscope.

5. ABOUT THE AUTHOR

Edward Ball is a creative technologist based in London. A physics graduate, web developer and hobbyist musician, he enjoys blending these three domains together via side projects such as academo.org and outputchannel.com.

6. REFERENCES

[1] mohayonao. Stereo analyser node. https://github.com/mohayonao/stereo-analyser-node, 2016.

[2] E. W. Weisstein. Hypotrochoid. From MathWorld—A Wolfram Web Resource. Last visited on 14/4/2017.

[3] E. W. Weisstein. Lissajous curve. From MathWorld—A Wolfram Web Resource. Last visited on 14/4/2017.

[4] E. W. Weisstein. Rose. From MathWorld—A Wolfram Web Resource. Last visited on 14/4/2017.